COVID-19 Outcome Prediction

Name:- Amany Azzam

ID:- 20399133

Dr. Marwa Elsayed

Eng. Amr Zaki

Project Description

- The data used in this project will help to identify whether a person is going to recover from coronavirus symptoms or not based on some pre-defined standard symptoms.

- These symptoms are based on guidelines given by the World Health Organization (WHO).This dataset has daily level information on the number of affected cases, deaths and recovery from 2019 novel coronavirus.

- The data is available from 22 Jan, 2020.

Data pre-processing

- Show the shape of the dataset . And found that consist of 863 rows and 14 columns.
- The columns consist of 13 features and 1 label or target.
- Checking for null values.
- Checking Data-type of each column.

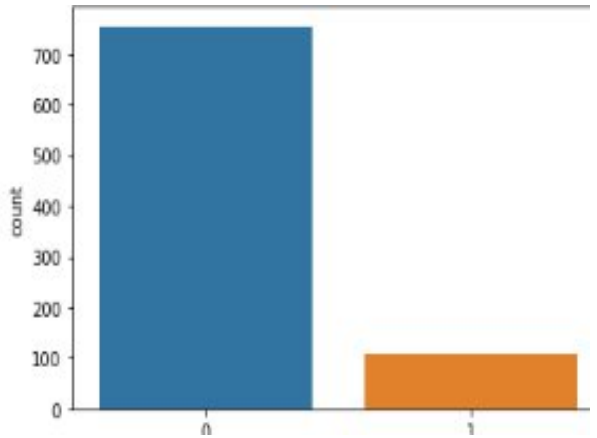
```
print("Size/Shape of the dataset: ",covid.shape)
print("Checking for null values:\n",covid.isnull().sum())
print("Checking Data-type of each column:\n",covid.dtypes)
```

Data pre-processing (cont.)

- Count the 1’s and 0’s in target column
- Now we have a problem that the two classes not balance

```
counter = Counter(y)
print(counter)
```

```
Counter({0: 755, 1: 108})
```

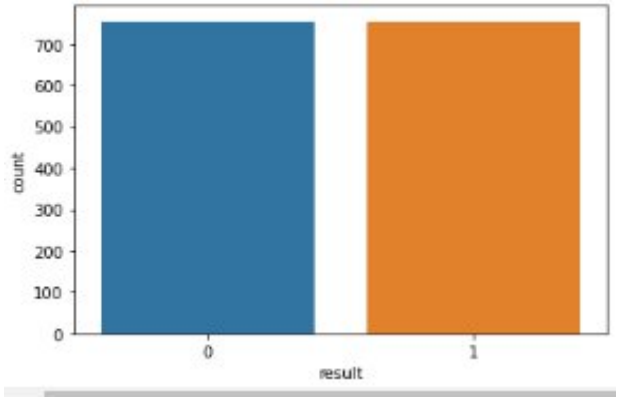


- We can use Smote to solve this problem and resample data to make it balance

```
oversample = SMOTE(k_neighbors=2)
X_train, y_train = oversample.fit_resample(X, covid["result"])
```

```
counter = Counter(y_train)
print(counter)
```

```
Counter({1: 755, 0: 755})
```



Data pre-processing (cont.)

- Split data into train and test by 80% for training and 20% for

```
X_train, X_test, y_train, y_test = train_test_split(X_train,y_train , test_size=0.2,random_state=25 , shuffle=True)
```

- Rescale the value of data by using StandardScaler

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

How choice hyper parameter

- I used GridSearchCV to find optimal hyperparameters for each classifier.

- GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model

K-Nearest Neighbors

Best Hyper-parameter for KNN

```
bests = grid_search.best_estimator_
best_li = grid_search.best_params_
print(bests)
print(best_li)
```

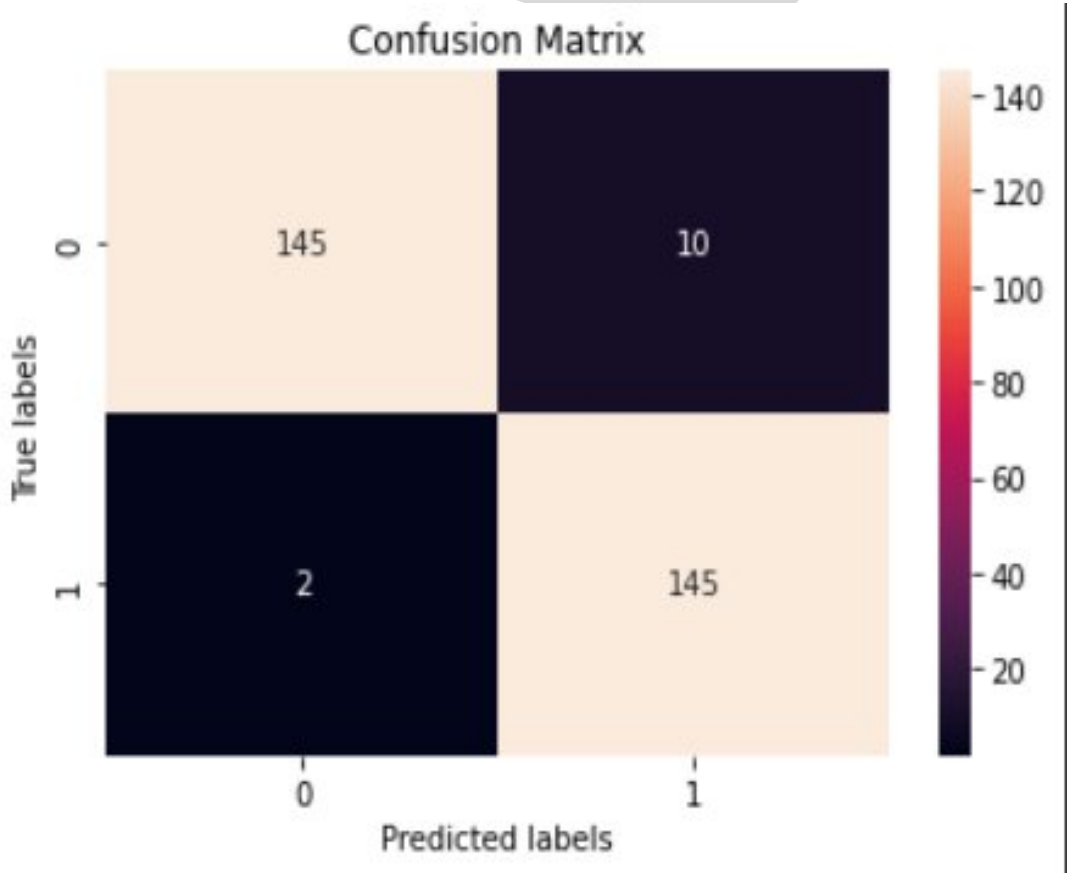
```
KNeighborsClassifier(n_neighbors=3, p=1, weights='distance')
{'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
```

Classification Report for KNN

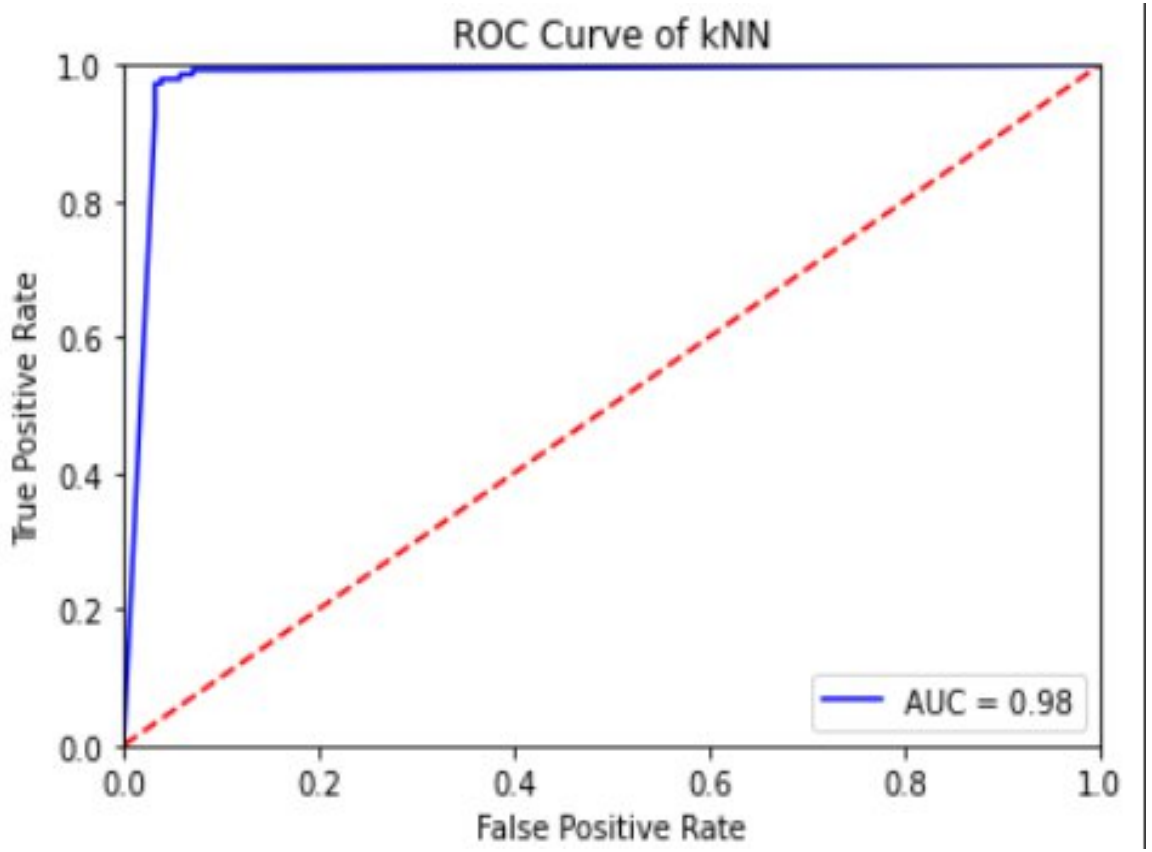
```
Classification Report is:
              precision    recall  f1-score   support

           0       0.99      0.94      0.96       155
           1       0.94      0.99      0.96       147

    accuracy                           0.96       302
   macro avg       0.96      0.96      0.96       302
weighted avg       0.96      0.96      0.96       302


 F1:
 0.9602649006622516

 Precision score is:
 0.9354838709677419

 Recall score is:
 0.9863945578231292
Model accuracy score: 0.9603
```

K-Nearest Neighbors

Confusion Matrix for KNN



AUC/ROC for KNN



Logistic Regression

Best Hyper-parameter for Logistic

```
bests = grid_search.best_estimator_
best_li = grid_search.best_params_
print(bests)
print(best_li)
```

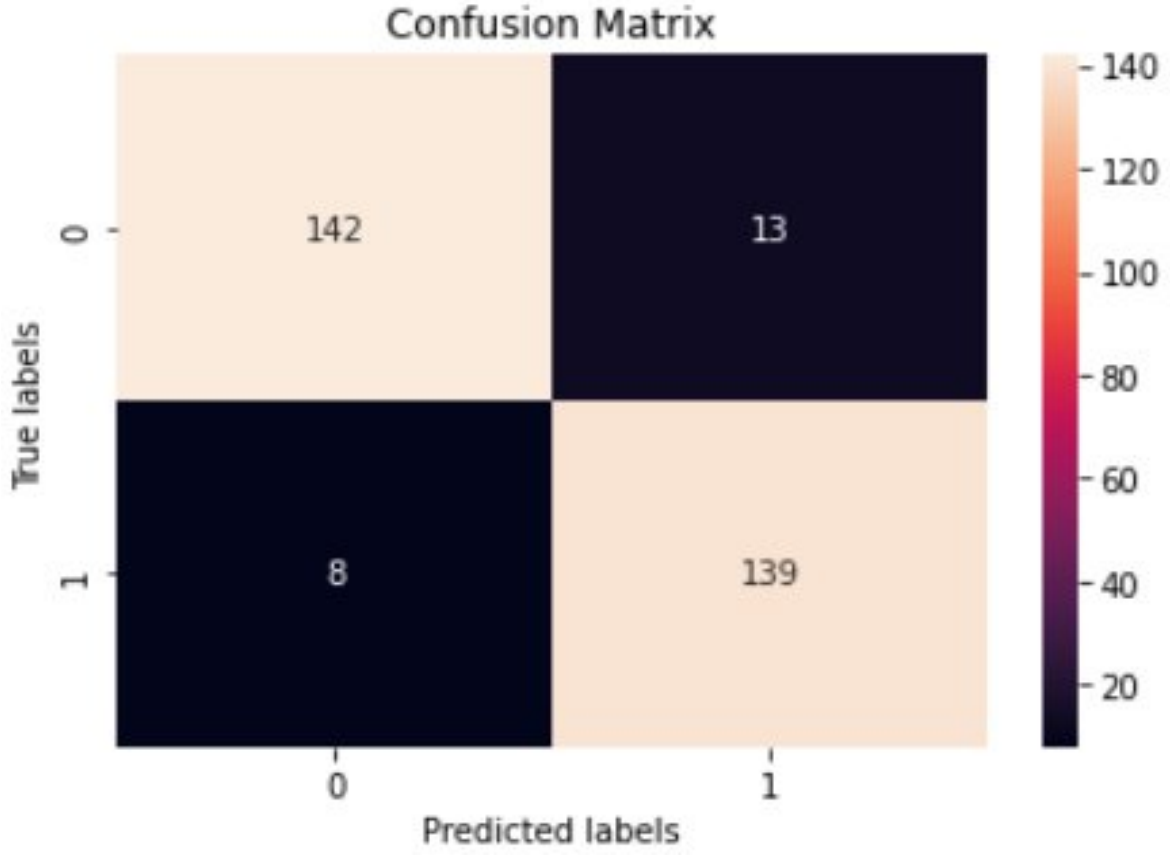
```
LogisticRegression(C=0.09, penalty='l1', solver='liblinear')
{'C': 0.09, 'penalty': 'l1', 'solver': 'liblinear'}
```

Classification Report for Logistic

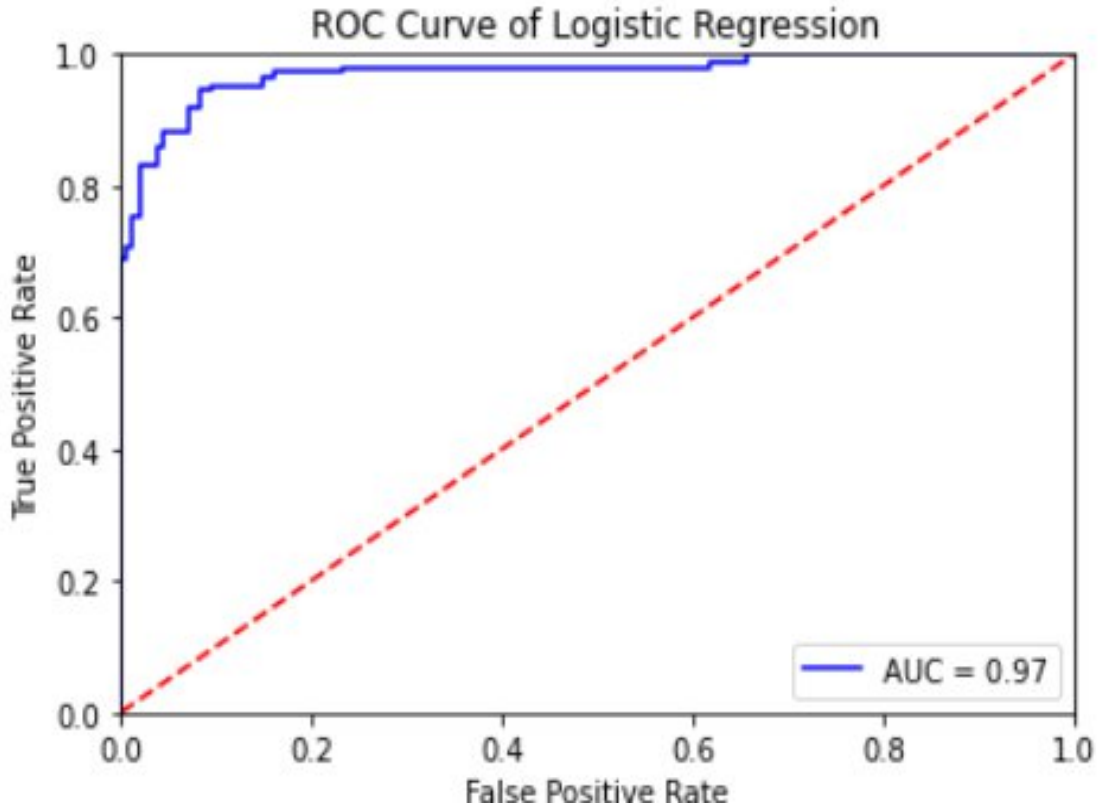
```
Classification Report is:
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       155
           1       0.91      0.95      0.93       147

    accuracy                           0.93       302
   macro avg       0.93      0.93      0.93       302
weighted avg       0.93      0.93      0.93       302


 F1:
 0.9297658862876254

 Precision score is:
 0.9144736842105263

 Recall score is:
 0.9455782312925171
Model accuracy score: 0.9305
```

Logistic Regression

Confusion Matrix for Logistic



AUC/ROC for Logistic



Naïve Bayes

Best Hyper-parameter for Naive Bayes

```
bests = grid_search.best_estimator_
best_li = grid_search.best_params_
print(bests)
print(best_li)
```

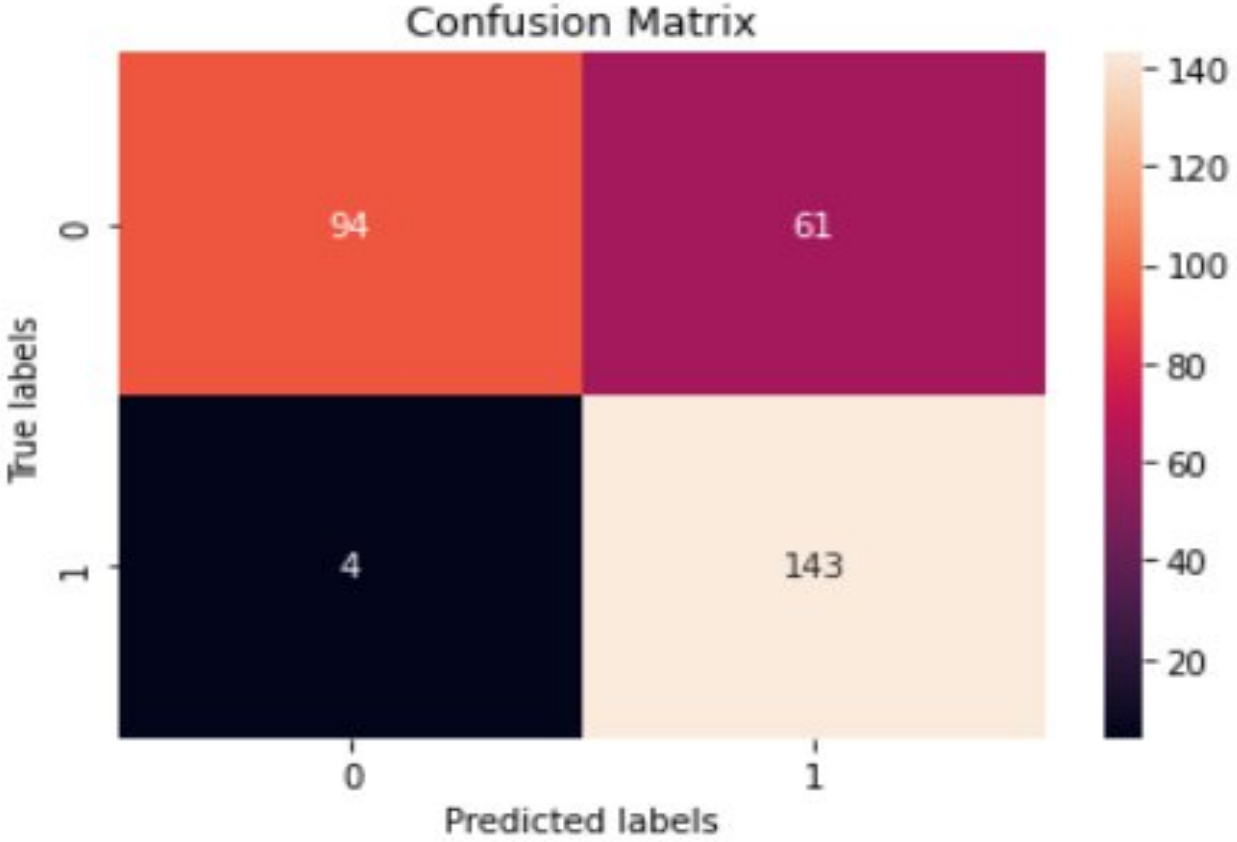
```
GaussianNB(var_smoothing=1)
{'var_smoothing': 1}
```

Classification Report for Naive Bayes

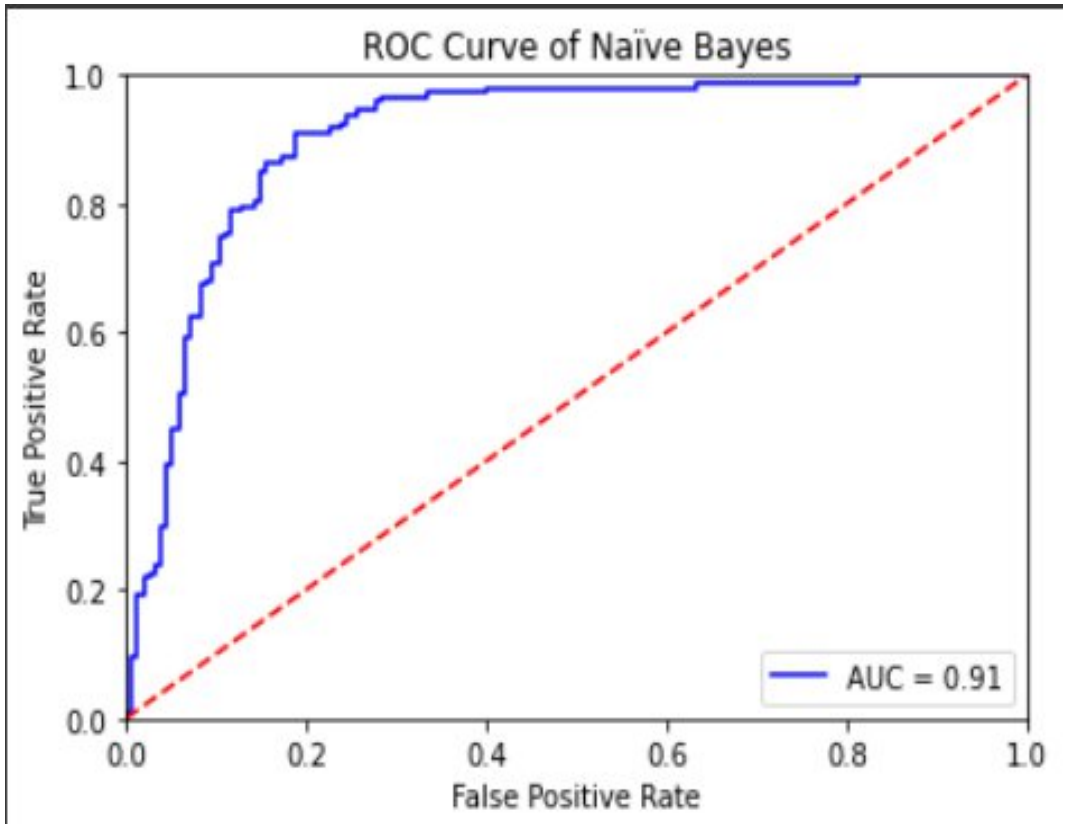
```
Classification Report is:
              precision    recall  f1-score   support

           0       0.96      0.61      0.74       155
           1       0.70      0.97      0.81       147

    accuracy                           0.78       302
   macro avg       0.83      0.79      0.78       302
weighted avg       0.83      0.78      0.78       302


 F1:
 0.8148148148148148

 Precision score is:
 0.7009803921568627

 Recall score is:
 0.9727891156462585
Model accuracy score: 0.7848
```

Naïve Bayes

Confusion Matrix for Naive Bayes



AUC/ROC for Naive Bayes



Decision Trees

Best Hyper-parameter for Decision Trees

```
bests = grid_search.best_estimator_
best_li = grid_search.best_params_
print(bests)
print(best_li)
```

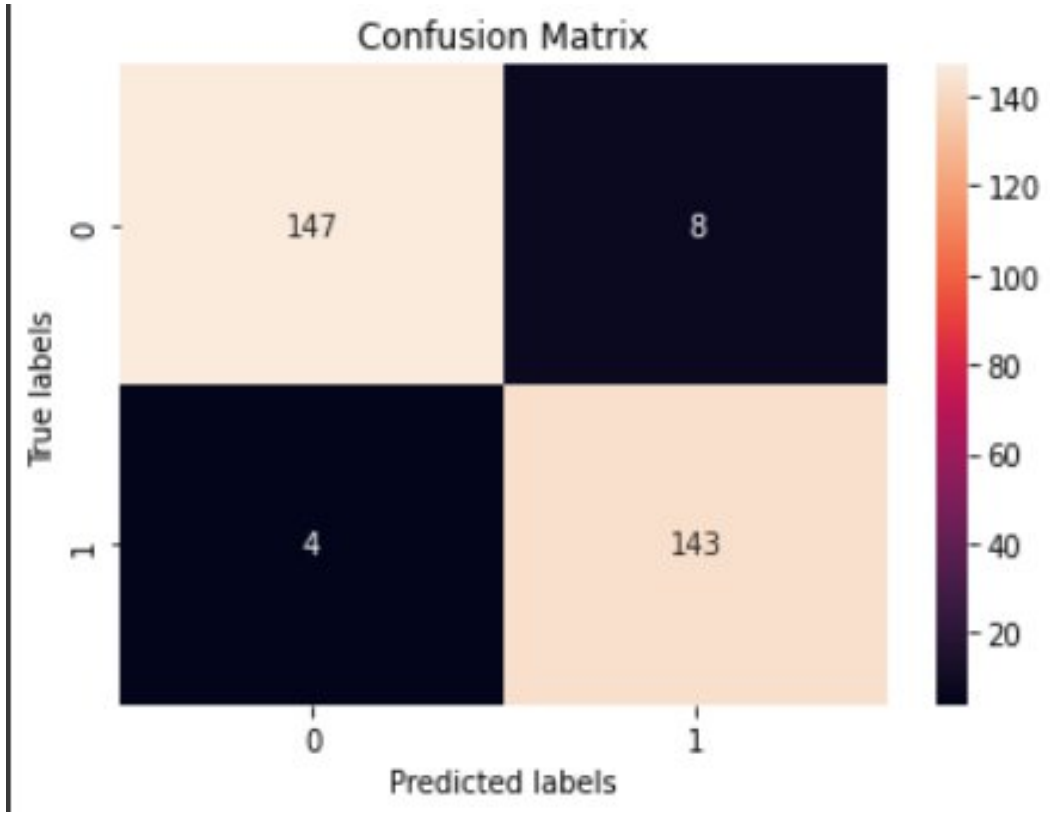
```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=10, random_state=3)
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 10}
```

Classification Report for Decision Trees

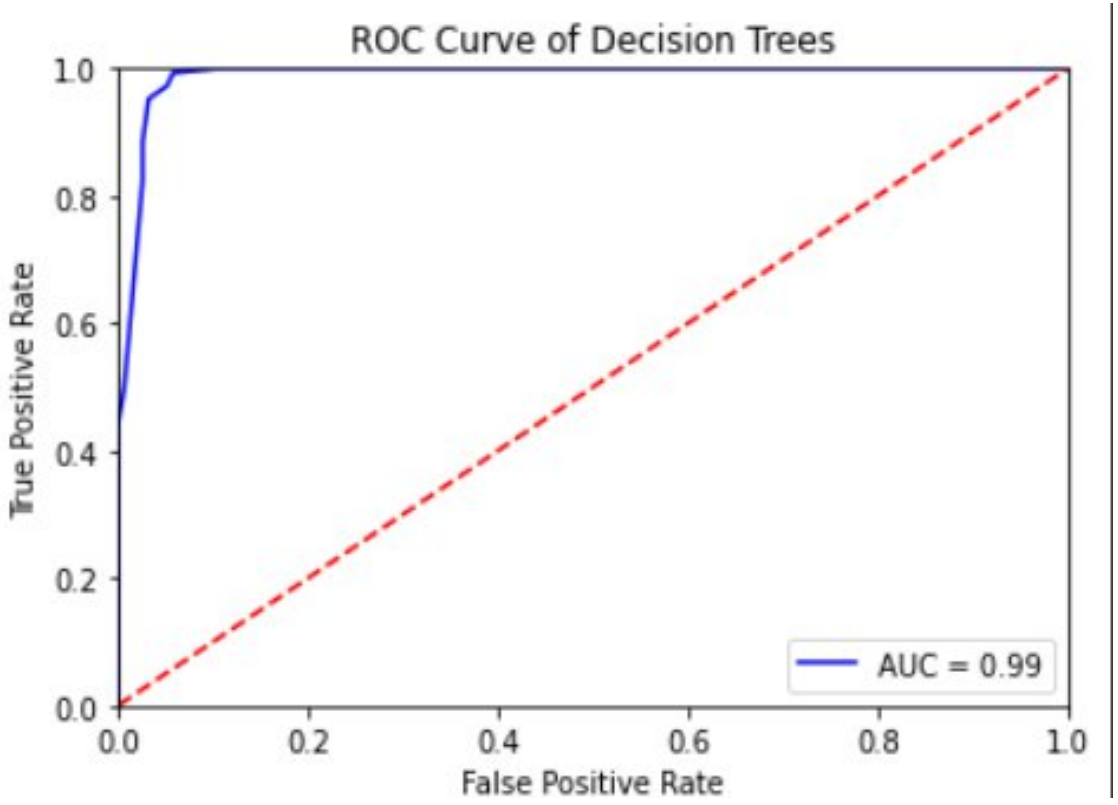
```
Classification Report is:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       155
           1       0.95      0.97      0.96       147

    accuracy                           0.96       302
   macro avg       0.96      0.96      0.96       302
weighted avg       0.96      0.96      0.96       302


 F1:
 0.959731543624161

 Precision score is:
 0.9470198675496688

 Recall score is:
 0.9727891156462585
Model accuracy score: 0.9603
```

Decision Trees

Confusion Matrix for Decision Trees



AUC/ROC for Decision Trees



Support Vector Machines

Best Hyper-parameter for SVC

```
bests = grid_search.best_estimator_
best_li = grid_search.best_params_
print(bests)
print(best_li)
```

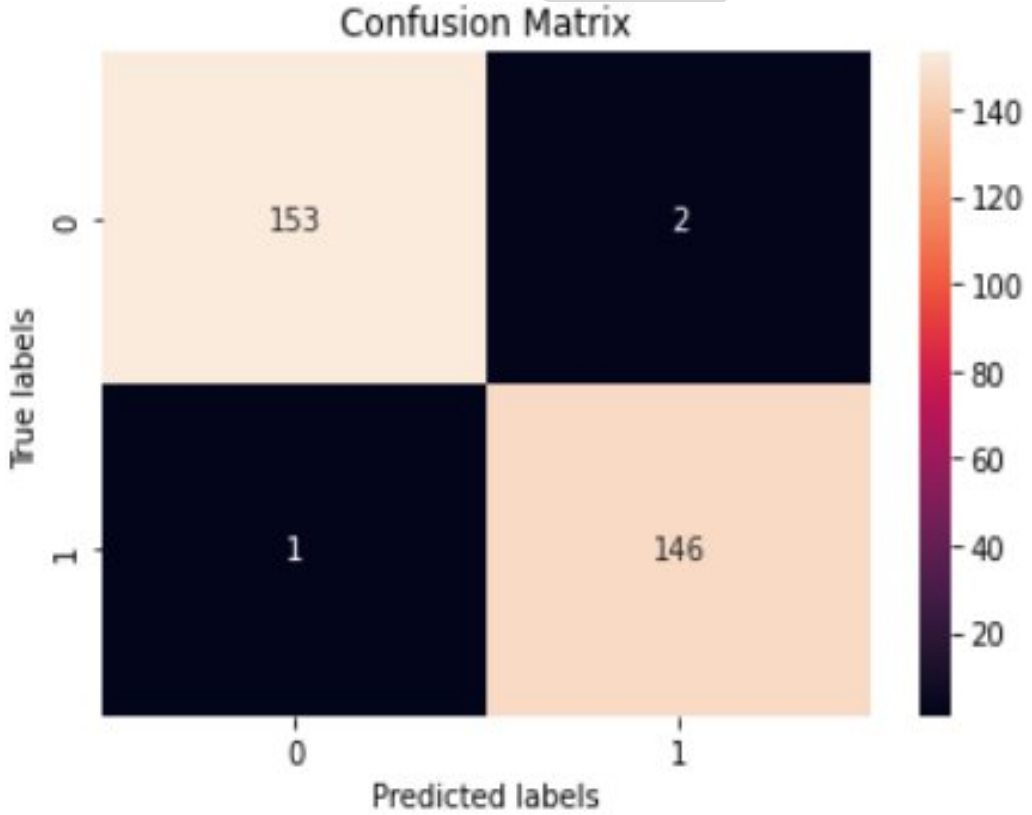
```
SVC(C=100, gamma=0.1, probability=True, random_state=3)
{'C': 100, 'gamma': 0.1, 'kernel': 'rbf', 'probability': True}
```

Classification Report for SVC

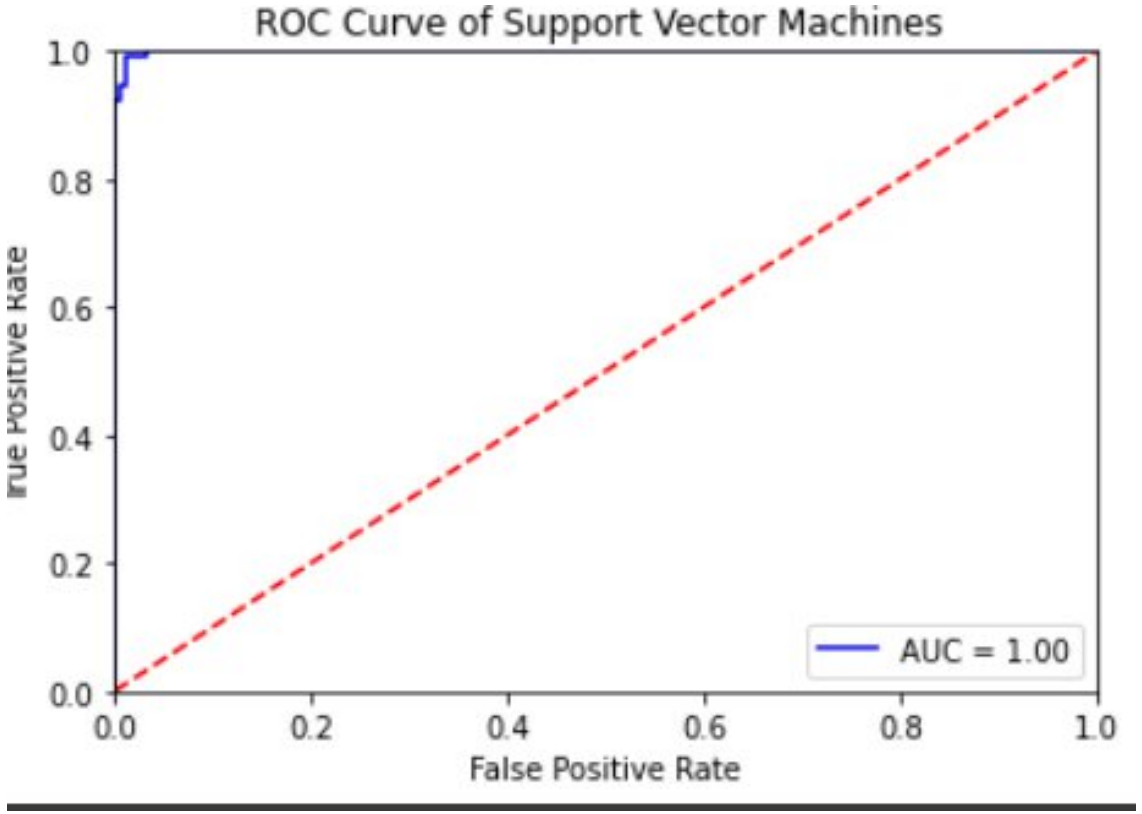
```
Classification Report is:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       155
           1       0.99      0.99      0.99       147

    accuracy                           0.99       302
   macro avg       0.99      0.99      0.99       302
weighted avg       0.99      0.99      0.99       302


 F1:
 0.9898305084745763

 Precision score is:
 0.9864864864864865

 Recall score is:
 0.99319727891156646
Model accuracy score: 0.9901
```

Support Vector Machines

Confusion Matrix for SVC



AUC/ROC for SVC



	KNN	Logistic	Naive Bayes	Decision Tree	SVC
Accuracy	0.9603	0.9205	0.7848	0.9603	0.9901
Recall	0.9863	0.9455	0.9727	0.9727	0.9931
Precision	0.9354	0.8967	0.7009	0.94701	0.9864
F1	0.9602	0.9205	0.8148	0.9597	0.9898
ROC/AUC	0.98	0.97	0.91	0.99	1.00

THANK YOU